- # Advantra User Authentication

Advantra Technology is an Edtech startup building a simple user authentication system for its web application. The system needs to allow new users to register, existing users to log in, and administrators to remove users when necessary. For now, the system will be managed entirely in memory using Python dictionaries, without a database.

Problem Statement

You have been hired as a Python developer to implement and develop the authentication system. The minimum implementation should support:

- **Login**: Checking if a user exists and verifying their password.
- **Register**: Adding a new user with validation checks (valid email, strong password, no duplicates).
- **Remove User**: Deleting a user by ID (partially implemented, with a duplicate function definition).

Data Structure

Users must be stored in a dictionary with the following structure:

```
users = {
   id: { # Integer
      "email": "example@address.com",
      "name": "John Doe",
      "password": "password",   # plaintext for now (can be improved later)
      "is_admin": False         # default is False
   }
}
```

Functions to Implement

1. register(email: str, name: str, password: str) -> dict

- Arguments:

- *email: user's email address*
- *name: user's full name*
- *password: chosen password*

- Validations:

- Email must contain '@'.
- Password must be at least 8 characters long and contain at least one digit.
- Email must be unique (no duplicate users).

- **Action**:

- If valid, create a new user with a unique id.
- Default 'is_admin': False.

- Return:

- {"status": "success", "message": "User registered successfully", "user_id": 2}
- {“status”: “error”, “message”: “User email already taken”}

2. login(email: str, password: str) -> dict

- Arguments:

- email: user’s email
- password: user’s password

- Action:

- Check if user exists and if password matches.

- Return:

- {"status": "success", "message": "Login successful", "user_id": 2}
- {"status": "error", "message": "Invalid email or password"}

3. remove_user(user_id: int) -> dict

- Arguments:

- user_id: the unique ID of the user to be removed

- Action:

- - If user exists, remove them.

- Return:

- {"status": "success", "message": "User removed successfully"}

- {"status": "error", "message": "User not found"}

Bonus Challenge

1. Implement list_users() → returns all users' id, email, and name (but never passwords).
2. Implement update_password(user_id: int, old_password: str, new_password: str) → validates the old password, applies the same security checks, and updates if valid.